INFORMATION AND COMPUTATION 89, 133–143 (1990)

Turing Machines with Access to History

BOGDAN S. CHLEBUS

Instytut Informatyki, Uniwersytet Warszawski, PKiN p. 850, 00-901 Warszawa, Poland

We study *remembering* Turing machines, that is Turing machines with the capability to access freely the history of their computations. These devices can detect in one step via the oracle mechanism whether the storage tapes have exactly the same contents at the moment of inquiry as at some past moment in the computation. The $s(n)$-space-bounded remembering Turing machines are shown to be able to recognize exactly the languages in the time-complexity class determined by bounds exponential in $s(n)$. This is proved for deterministic, non-deterministic, and alternating Turing machines. © 1990 Academic Press, Inc.

1. INTRODUCTION

We investigate the computational power of Turing machines augmented with the capability of accessing freely the history of their computations, without the necessity of storing any record of it. We call such Turing machines the *remembering Turing machines* (R-TMs, in short). Technically R-TMs resemble oracle TMs. They have a special query state distinguished, and if an R-TM enters this state then in the next step it enters one of two other special states, depending on whether or not, respectively, the symbols written on the tapes are exactly the same as those in a certain past configuration.

The main result of this paper is a relationship between the time- and space-bounded R-TMs and the respective ordinary TMs. First we show that the time-bounded remembering and ordinary TMs are polynomially related. On the other hand, it turns out that the $s(n)$-space-bounded R-TMs recognize the languages in the time complexity classes determined by bounds $c^{s(n)}$, for $c > 1$. This result holds, respectively, for deterministic, non-deterministic, and alternating remembering TMs (DR-TMs, NR-TMs, and AR-TMs, in short). We show that there are possible certain restrictions on the NR-TMs and AR-TMs which preserve the exponential relationship between the space-bounded R-TMs and the time-bounded ordinary TMs. The NR-TMs can be restricted to have the "no"-state (that is one of the special states) as the only final rejecting state, and the AR-TMs can be restricted to be able to enter the query state at most once in every computation.

The alternating TMs with either absolute or conditional restrictions on looping were previously investigated by Robson (1985). One of the specific models he studied, the *non-looping* ATM, was defined so that computations were considered valid only if they did not contain repetitions of configurations. For the non-looping ATMs Robson (1985) proved a theorem analogous to the part of Theorem 3.1 concerning alternating remembering TMs (actually he proved a stronger result described as Proposition 3.7 in this paper). We generalize his result to the deterministic and non-deterministic TMs. The proof given in this paper covers all three kinds of TMs. It is simple and intuitive, which is mainly due to domino tilings being unsed instead of direct manipulations with Turing machines. A key distinction in our approach to the very definition of a "history-dependent" mode of computations is the presence of the oracle mechnism. This is what actually enables us to carry over the generalization in the deterministic case. In the non-deterministic case Theorem 3.1 can be proved for non-looping TMs, as is shown in Section 5.

The AR-TM has a natural interpretation as a computing device modelling games in which the "value" of a position, for example, whether it is a winning position or not, depends on the previous positions entered in the course of the play (cf. Lichtenstein and Sipser, 1980; Robson, 1984). Such history-dependent games are formally neither perfect nor partial-information games. In this way the theory of the non-looping ATMs and the remembering ATMs is a contribution to the program initiated by Chandra, Kozen, and Stockmeyer (1981), who introduced alternation as a mode of computation modelling the perfect-information games. Then Peterson and Reif (1979) and Reif (1979) introduced other "alternations," namely the *multiple-person* one (Peterson and Reif, 1979) and the *private and blindfold* ones (Reif, 1979) (see also Jones, 1978). The underlying games in these new alternations are of partial information, and in the multiple-person alternation there are moreover two teams of players rather then two single players (Peterson and Reif, 1979). Recently Ladner and Norman (1985) studied the *solitaire automata*, that is such private-alternation TMs in which the game is started by player 0 (the player who aims at rejection of the input) and once player 1 enter the game, player 0 acts deterministically. The main results of the papers cited above concerned mutual relationships between the complexity classes of the various alternations and the deterministic and non-deterministic classes.

2. Definitions

In this section we define the R-TMs and recall certain facts concerning domino-tiling problems.

2.1. *Remembering Turing Machines*

The R-TM M is a Turing machine M with three special states: the *query state* q_I, the *"yes"-state* q_y, and the *"no"-state* q_N. From q_I M enters either q_Y or q_N, depending on whether or not, respectively, there has been before a configuration in the computation identical to the present configuration except for the state and the head positions. We say that an *R-loop* has been detected if M is in q_Y. The R-TMs are considered in three versions: the deterministic, non-deterministic, and alternating ones. The respective semantics of their computations are natural extensions of those of the original models (cf. Chandra *et al.* 1981; Hopcroft and Ullman, 1979). Alternation in the R-TMs can be conveniently interpreted as a game process in the following standard way (see Chandra *et al.*, 1981; Chlebus, 1986; Grädel, 1987; Ladner and Norman, 1985; Peterson and Reif, 1979; Reif, 1979; Robson, 1985). There are two players: the *existential player* (of preference) whose aim is to lead the TM to acceptance, and the *universal player* who aims at rejection of the input. When an AR-TM M enters an existential state, then the next move is decided by the existential player, and when M is in a universal state, the next move is chosen by the universal player. The query state is neither existential nor universal. The existential player wins the game if M enters an accepting state. We refer to this game as the *M-game*. Given an input x, M *accepts* x iff there is a winning strategy for the existential player in the M-game starting at the initial configuration with input x. The *time* of a winning strategy S is the least upper bound on the times of the plays against all the strategies of the universal player, when the existential player employs S. M is said to *accept within time* $t(n)$ if, for each input x of length n, there is a winning strategy S such that the time of S on input x is not greater than $t(n)$. The *space* bounds are defined analogously. The time- and space-bounded deterministic and non-deterministic R-TMs are defined in the standard way (see Hopcroft and Ullman, 1979). The complexity classes DR-TIME($t(n)$), NR-TIME($t(n)$), and AR-TIME($t(n)$) are the classes of languages recognized by the DR-TMs, NR-TMs, and AR-TMs, respectively, in time $t(n)$. The space complexity classes DR-SPACE($s(n)$), NR-SPACE($s(n)$), and AR-SPACE($s(n)$) are defined analogously. We use the following notations: if X denotes one of the letters D, N, or A, then XR-TIME(EXP($f(n)$)) $= \bigcup_{c>0} X$R-TIME($c^{f(n)}$); the class XR-SPACE(EXP($f(n)$)) is defined similarly.

2.2. *Bounded Domino-Tiling Problems*

The key parts of the proofs of the facts relating time- and space-bounded R-TMs and the ordinary TMs, which are described in the sequel, use certain results concerning bounded domino-tiling problems. A *domino type* is a quadruple of colours. A unit square with edges coloured as specified by the type is called a *domino tile* of the given type. Domino tiles cannot be rotated or inverted. Suppose there is given a finite set T of domino types and a natural number m. A *T-tiled $m \times m$-square* is an $m \times m$-square grid covered by dominoes of types in T in such a way that each pair of adjacent tiles has the same colour along their common boundary, and the external boundaries are coloured some fixed colour, say, white (cf. Chlebus, 1986, 1987; Grädel, 1987; Harel, 1983; Lewis and Papadimitriou, 1981; Savelsberg and van Emde Boas, 1984). There is a close relationship between tilings and Turing-machine computations. An attempt to formalize this relationship was undertaken in (Chlebus, 1985). Sometimes it is more convenient to deal with domino tilings rather than directly with Turing machines, and the proofs of Lemma 3.5, Proposition 3.6, and Proposition 3.7 may serve as examples. Below we describe in detail the relevant facts concerning alternating TMs and domino-tiling games; similar facts relating the deterministic and non-deterministic TMs and the existential versions of domino-tiling problems follow similarly from the results described, for example, by Savelsberg and van Emde Boas (1984) (see also Chlebus, 1985, 1987; Harel, 1983; Lewis and Papadimitriou, 1981).

Consider the following tiling game, for a given set T of domino types and a natural number m. There are two players: CONSTRUCTOR and SABOTEUR. They cover an $m \times m$-square grid with dominoes of types in T in the course of a game. The moves are performed alternately, one tile is placed at a time, and CONSTRUCTOR starts the game. First the bottom row is filled from left to right, then the second one in the same order, and so on. A move is legal provided the selected tile has matching colours along their common boundaries with the tiles adjacent to the left (if any) and to below (if any). If after completing this process a T-tiled $m \times m$-square is obtained then CONSTRUCTOR wins. In the other case, that is if either a deadlock occurs and no legal move could have been performed or the external side of a tile placed on the border part of the grid was not white, then SABOTEUR wins. It was shown by Chlebus (1986) that for each alternating single-tape TM M there are sets of domino types such that the accepting computations of M correspond to the bounded tilings obtained during the tiling game. More precisely, there is a finite set T_M of domino types such that for each word x over the input alphabet of M there is a set T_x of domino types with the following properties:

(a) M accepts x in time $t(|x|)$ iff CONSTRUCTOR has a forced win in the $2t(|x|) \times 2t(|x|)$-square $T_M \cup T_x$-tiling game;

(b) The bottom row of any $T_M \cup T_x$-tiled $k \times k$-square, for $k > |x|$, is unique and all the tiles occurring in it are from T_x (they encode the initial configuration of M on x by the colours on their top boundaries; the tiles from T_x are used only in the bottom row).

Other variants of domino-tiling games were studied by Grädel (1987, 1988).

3. Time- and Space-Bounded Remembering Turing Machines

The relationships between time- and space-bounded R-TMs and their non-remembering counterparts are summed up in the following theorem (cf. Robson (1985) in the case of alternation).

3.1. Theorem. *Let X denote one of the letters D, N, or A.*

(a) XR-TIME(EXP($t(n)$)) $=$ XTIME(EXP($t(n)$)), *provided* $t(n) \geqslant n$ *and $t(n)$ is time constructible.*

(b) XR-SPACE($s(n)$) $=$ XTIME(EXP($s(n)$)), *provided* $s(n) \geqslant \log n$ *and $s(n)$ is space constructible.*

Part (a) of this theorem for X equal to A can be strengthened to the equality AR-TIME($t(n)$) $=$ ATIME($t(n)$)—see Robson (1985) for a proof. The alternating time and space complexity classes can be expressed as deterministic ones as was showed by Chandra, Kozen, and Stockmeyer (1981). After converting Theorem 3.1, for $X = A$, into the relationships between the alternating-remembering and deterministic complexity classes, one obtains the same relationships as Reif (1979) proved for the blindfold alternation.

The proof of Theorem 3.1 is divided into four parts corresponding to the respective four inclusions.

3.2. Lemma. XR-TIME($t(n)$) $\subseteq$ XTIME($t^3(n)$), *provided* $t(n) \geqslant n$ *and $t(n)$ is time constructible.*

Proof. Consider an XR-TM M operating in time $t(n)$. The X-TM M_1 simulates M as follows. M_1 stores the previous configurations of M in consecutive blocks of length $t(n)$ on its tapes. To perform an ordinary move of M, M_1 lays off $t(n)$ new cells on the tapes, and then decides the move and stores the next configuration in the allotted space. To handle a move from the query state, M_1 scans the previous configurations. The simulation of both an ordinary and a query move takes $O(t^2(n))$ of time. The overall time is this $O(t^3(n))$. ▮

3.3. Lemma. XTIME($t(n)$) $\subseteq$ XR-TIME($t(n)$), *provided* $t(n) \geqslant n$.

Proof. Observe that XTMs are XR-TMs which never enter their query states. ▌

Part (a) of Theorem 3.1 follows from these two lemmas.

3.4. Lemma. XR-SPACE$(s(n)) \subseteq X$TIME$(c^{s(n)})$ *for a certain* $c > 1$, *provided* $s(n) \geqslant \log n$ *and* $s(n)$ *is space constructible.*

Proof. Let M be an XR-TM operating in space $s(n)$. The XTM M_1 simulating M uses three storage tapes. The first tape simulates the storage tape of M. The second one stores the consecutive contents of the first tape of M_1 in the previous configurations, without repetitions. The third tape of M_1 stores a counter in the space $s(n)$ and with base d, where $d > 1$ is a certain number to be specified later on. To simulate a single step of M from a given configuration, M_1 decides the next configuration and updates the tapes. If this is an ordinary move of a TM, then it is handled in an obvious manner. A move from the query state is performed by first scanning the "history of computation" which is stored on the second tape, and then entering either q_Y or q_N accordingly. After having completed this M_1 checks if the second tape should be updated, that is whether the contents of the first tape are not stored anywhere on the second tape. If this is the case then M_1 lays off $s(n)$ new cells on the second tape and writes the contents of the first tape in this space. The counter on the third tape is incremented by 1 after completing a simulation of every move on M unless M enters q_N, in this case the counter is set to 1. The computation of M_1 starts with the input word being written on both the first and the second tape, and the counter set to 1. M_1 accepts if M enters an accepting state; M_1 rejects if either M rejects or the counter reaches its maximal value.

Let b be such a constant that there are no more than $b^{s(n)}$ configurations of M in space $s(n)$. It may happen that M needs to go through a number of configurations in order to continue the computation by entering the "yes"-oracle state when needed. Again it may happen that in order to reach such configurations a sequence of previously reached configurations should be repeated in the course of a computation. Observe that such a part of the computation contributing a new configuration must have length at most $b^{s(n)}$, and, since there are at most $b^{s(n)}$ distinct configurations, the number of such contributing computations is bounded by the same number. Therefore the number b^2 may be taken as d.

The simulation of a single step of M requires $O(d^{s(n)})$ steps of M_1. Therefore M_1 completes its computation within $d^{2s(n)}$ steps. ▌

3.5. Lemma. XTIME(EXP$(s(n))) \subseteq X$R-SPACE$(s(n))$, *provided* $s(n) \geqslant \log n$ *and* $s(n)$ *is space constructible.*

Proof. Let M be an XTM operating in time $c^{s(n)}$ for a $c > 1$. The required $s(n)$-space-bounded XR-TM M_1 equivalent to M simulates the process of constructing a suitable domino tiling. To describe M_1 we use the notations and properties concerning the bounded domino-tiling problems described in the previous section. Given and input x, denote by D_x the set $T_M \cup T_X$ of domino types, and by r_x the number $2c^{s(|x|)}$. M_1 has four storage tapes: t_0, t_1, t_2, and t_3. Tape t_0 stores a single element of D_x. Tapes t_1 and t_2 each store a natural number not greater than r_x. These numbers are equal to the column and row numbers, respectively, of the processed position on the tiling. More precisely, tapes t_1 and t_2 operate as c-ary counters in space $2s(|x|)$. The operation of "moving" to neighbouring positions in the tiling is performed by incrementing or decrementing the counters accordingly. Typical contents of the tapes of M_1 can be interpreted as follows: the domino stored on t_0 lies in the column with the number written on t_1 and the row with the number written on t_2 of the tiling being "built" in the course of the computation. M_1 begins its computation by writing the sequence of all the domino types in T_M on tape t_3. Positioning the head on such a type will be used as a means to remember it. (Note that T_M is a fixed set independent of the input.)

The tiling is constructed by selecting repeatedly first the tiles in the bottom row from left to right, then those in the second row in the same order, and so on. The selection of a consecutive domino is performed as follows. Suppose M_1 has just selected the tile, say, d_1 adjacent to the left of the processed vacant position, d_1 is stored on t_0, and tapes t_1 and t_2 contain the coordinates of d_1 in the tiling. M_1 begins by erasing t_0 and remembering d_1 by moving the head on tape t_3 to the position of type d_1. Then M_1 decrements the t_1-counter by 1, thus "moving" to the row below, and next increments the t_2-counter, thereby "moving" right to the next column. M_1 wants to get to know which domino tile has been placed at this position. To this end M_1 resorts to its power to remember the history. It repeatedly places consecutive dominoes on t_0 and asks the "oracle" whether the tapes looked exactly the same at some moment in the past. When an R-loop is detected eventually, then M_1 knows both the domino d_1 adjacent to the left, and the domino, say, d_2 adjacent to below the vacant position. M_1 selects a domino (if there is any) matching the right-side colour on d_1 and the top-side colour on d_2, places it on t_0, and increments the row counter by 1. After this step the next consecutive tile (adjacent to the right of the last position) is started to be selected in the same manner as that described above. If M is deterministic, then there is at most one tile fitting in the current vacant position; if M is non-deterministic, then M_1 chooses non-deterministically a proper domino; and finally if M is an alternating TM, then M_1 simulates the tiling game so that the moves of CONSTRUCTOR are being performed by the existential player, and the moves of SABOTEUR by the universal player.

It remains to describe the actions of M_1 when it is on the border part of the tiling, that is with one of the counters storing either 1 or r_x. The bottom row of tiles is uniquely determined: the first dominoes encoding the input are obtained through scanning the input, and the remaining part is filled with "blanks" up to the moment when the column counter on t_1 reaches its maximal value. Then the row counter on t_2 is incremented by 1 and the column counter on t_1 is set again to 1. This is always done when the right border is reached and the top one has not yet been reached. After a new domino on the border is selected then it is additionally verified to have the respective side (or sides) coloured white. If two counters are full and a proper domino completing a perfect tiling is selected then M_1 accepts.

The storage of M_1 is dominated by the counters which require $2s(n)$ cells. This can be easily decreased to exactly $s(n)$ cells (see Hopcroft and Ullman, 1979), thus providing the required space bound. ∎

This completes the proof of Theorem 3.1. Next we give two variants of this theorem. Proposition 3.6 could be proved by a modification of the proof of Lemma 3.5. We give another proof which demonstrates particular capabilities of NR-TMs. This proof is next adapted to alternation to yield Proposition 3.7.

3.6. PROPOSITION. *Theorem* 3.1 *holds true for non-deterministic remembering* TMs *if they are restricted to have the "no"-state q_N as the only rejecting final state.*

Proof. It is enough to prove Lemma 3.5 under this restriction. We need to show that there is a NR-TM M_1 constructing the domino tiling and having the "no"-state as the only rejecting state. M_1 is similar to the TM from the previous proof; the difference being that it does not have tape t_3. In the first stage M_1 guesses a tilling row by row, enforcing only the adjacency constraints along the vertical edges of the tiles. Selecting a new tile is performed by incrementing the column counter by 1 and writing such a new domino on t_0 which may be adjacent to the right of the one before. That last condition is guaranteed by the next-move relation of M_1. If the column counter reaches the maximal value then the row counter is incremented by 1, the column counter is set to 1, and a new row is started to be guessed. The boundary dominoes are also checked to have the respective external sides coloured white. After the first stage is completed, that is after r_x perfect rows have been constructed, M_1 performs the second stage. Now it tries to guess the same tiling in the column-by-column manner, enforcing the adjacency constraints along the horizontal boundaries by its next-move relation. Each time a new tile is decided, M_1 verifies whether it has been placed in this position during the first stage of the computation.

To this end M_1 enters the query state q_I. If M_1 goes into q_Y then M_1 proceeds to guess the next domino, else, if M_1 enters q_N, then M_1 rejects. M_1 accepts after the whole tiling has been guessed for the second time. ∎

3.7. PROPOSITION. *Theorem* 3.1 *holds true for the alternating-remembering* TMs *if they are restricted to be able to enter the query state q_I at most once in the course of a computation.*

Proof. Again it is enough to reprove Lemma 3.5. We use the same notations as those in the proofs of Lemma 3.5 and Proposition 3.6. The way the AR-TM M_1 simulates the domino-tiling game determined by D_x and r_x is similar to the first stage of computation of the M_1 from the previous proof. The tiling is guessed row by row, and the next-move relation of M_1 guarantees the proper adjacencies only along the vertical boundaries of the tiles. To enforce selections of tiles matching also the colours below, each of the players has the capability to control the moves of his adversary. If a player tries to "cheat" and places a tile with improper relationship to the tile below then the other player can win the M_1-game by proving that the move of the adversary was illegal. Technically this is done as follows. Suppose that one of the players has just selected a new domino tile and placed it on the tape t_0. The adversary can either challenge this selection or "skip" a verification and select a consecutive tile adjacent to the right. If he chooses to verify then he does the following: he selects a domino tile *not* matching below, writes it on t_0, decrements the row counter by 1, and centers q_I. From this state M enters either q_Y or q_N. If the former is the case then the player who has challenged wins the game, in the other case he loses it. Observe that if M enters q_Y then this is a proof that the player who selected the last domino above tried to cheat and so he ought to lose. If, on the other hand, M enters q_N, then it means that the player who decided to check has failed to provide such a proof. In this case the challenging player can be regarded as a loser since he did not have to verify selections of his adversary. M_1 at this moment is either in q_Y or in q_N and does not know who is the winner, but this can be found out by calculating whose turn it was to select a new tile, and hence who decided to verify the last selection. To carry out the calculations answering this question M_1 uses the values of the counters on t_1 and t_2. Knowing the answer and the fact of whether an R-loop has been detected or not, M_1 accepts of rejects accordingly. ∎

4. CONCLUSIONS

The main result of this paper, Theorem 3.1, is a generalization of a theorem of Robson (1985) relating non-looping ATMs to ordinary ATMs.

It concerns deterministic and non-deterministic TMs. To carry out the generalization an apparently more flexible model of a TM capable of resorting to the history of computation by way of an oracle was introduced. The viability of this model is demonstrated in the proofs of theorems in Section 3.

There is a natural question whether an analogue of Theorem 3.1 holds true for the deterministic and non-deterministic non-looping TMs (defined along the lines of Robson (1985)). The answer in the case of DTMs is negative since the ordinary and non-looping variants are clearly equivalent. It turns out that Theorem 3.1 holds true for non-looping NTMs. The most difficult part is Lemma 3.5. To prove it in a similar way it is enough to have a mechanism by which a non-looping NTM can verify that a particular tile type was placed at a particular grid position. To this end it is sufficient to verify that every other tile type was not placed there. This verification can be carried out by a non-looping NTM.

R-TMs also can be interpreted as computing devices having an auxiliary space to store their history such that both the amount of it and the time to scan it are "for free." A similar idea of studying computations capable of resorting to "hidden" resources was previously explored by Mager (1969) and Cook (1971). They investigated the auxiliary pushdown automata (APDAs, in short), that is TMs which have an auxiliary memory organized as a pushdown store. The space complexity of such devices is defined by the amount of the general-purpose storage used, while the stack space is not counted. Cook (1971) proved that deterministic and non-deterministic APDAs with the same space bounds are equivalent, and moreover that $\bigcup_{c>0} \mathrm{DTIME}(c^{s(n)})$ is exactly the class of languages recognized by APDAs using space $s(n)$. It follows from Theorem 3.1 that both the APDAs and DR-TMs with $s(n)$-space bound recognize the same class of languages.

Acknowledgments

Thanks are due to Marek Chrobak for his encouragement during this research, and to the anonymous referee for pointing out the equivalence of NR-TMs and non-looping NTMs.

Received April 28, 1988; final manuscript received July 7, 1989

References

Chandra, A. K., Kozen, D. C., and Stockmeyer, L. J. (1981), Alternation, *J. Assoc. Comput. Mach.* **28**, 114–133.

Chlebus, B. S. (1985), From domino tilings to a new model of computation, *in* "Lecture Notes in Computer Science," Vol. 208, p. 24–33, Springer-Verlag, Berlin.

CHLEBUS, B. S. (1986), Domino-tilling games, *J. Comput. System Sci.* **32**, 374–392.

CHLEBUS, B. S. (1987), Proving NP-completeness using BOUNDED TILING, *J. Inform. Process. Cybernet. EIK* **23**, 479–484.

COOK, S. A. (1971), Characterizations of pushdown acceptors in terms of time bounded computers, *J. Assoc. Comput. Mach.* **18**, 4–18.

GRÄDEL, E. (1987), "The Complexity of Subclasses of Logical Theories," Dissertation, Basel.

GRÄDEL, E. (1988), Domino games, with an application to the complexity of Bolean algebras with bounded quantifier alternation, *in* 'Proceedings, Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, Vol. 294, pp. 98–107. Springer-Verlag, Berlin.

HAREL, D. (1983), Recurring dominoes: Making the highly undecidable highly understandable, *in* "Lecture Notes in Computer Science," Vol. 158, pp. 177–194, Springer-Verlag, Berlin; Effective transformations on infinite trees, with applications to high undecidability, dominoes and fairness, *J. Assoc. Comput. Mach.* **33** (1986), 224–248.

HOPCROFT, J. E., AND ULLMAN, J. D. (1979), "Introduction to Automata Theory, Languages and Computation," Addison–Wesley, Reading, MA.

JONES, N. D. (1978), Blindfold games are harder than games with perfect information, *Bull. EATCS* **6**, 4–7.

LADNER, R. E., AND NORMAN, J. K. (1985), Solitaire automata, *J. Comput. System Sci.* **30** 116–129.

LEWIS, H. R., AND PAPADIMITRIOU, C. H. (1981), "Elements of the Theory of Computation," Prentice–Hall, Englewood Cliffs, NJ.

LICHTENSTEIN, D., AND SIPSER, M. (1980), Go is polynomial-space hard, *J. Assoc. Comput. Mach.* **27**, 393–401.

MAGER, G. (1969), Writing pushdown acceptors, *J. Comput. System Sci.* **3**, 276–319.

PETERSON, G. L., AND REIF, J. H. (1979), Multiple-person alternation, *in* "Proceedings, 20th Symposium on Foundations of Computer Science," pp. 348–363, IEEE Computer Society, New York.

REIF, J. H. (1979), Universal games of incomplete information, *in* "Proceedings, 11th ACM Symposium on Theory of Computation," pp. 288–308, Assoc. for Comput. Mach., New York; The complexity of two-player games of incomplete information, *J. Comput. System Sci.* **29** (1984), 274–301.

ROBSON, J. M. (1984), *N* by *N* checkers is exptime complete, *SIAM J. Comput.* **13**, 252–267.

ROBSON, J. M. (1985). Alternation with restrictions on looping, *Inform. and Control* **67**, 2–11.

SAVELSBERG, M. P. W., AND VAN EMDE BOAS, P. (1984), BOUNDED TILING, an alternative to SATISFIABILITY? *in* "Proceedings, 2nd Frege Conference" (G. Wechsung, Ed.), Akademie Verlag, Mathematische Forschung 20, pp. 354–363